PRIME XML Automation User Guide

Version 4.10.10

September 2025

SINCE DAY ONE.

Chyron PRIME XML Automation User Guide • 4.10.10 • September 2025 • This document is distributed by Chyron in online (electronic) form only, and is not available for purchase in printed form.

Table of Contents

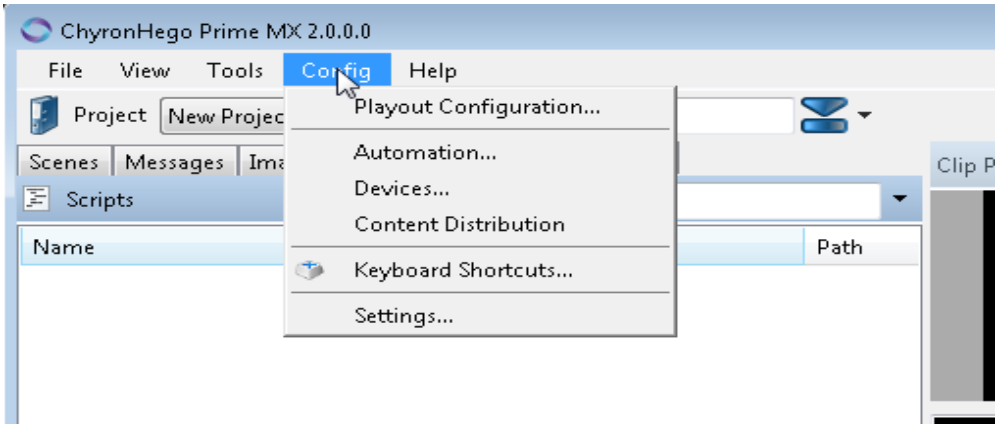
Introduction

The Prime XML automation control allows users to manipulate a currently running instance of the Prime 2.0 designer using predefined XML commands implemented in the Prime executable. The XML automation control makes use of a TCP connection between and Prime and an external application. The connection must be initialized in the Prime designer before making use of the automation control.

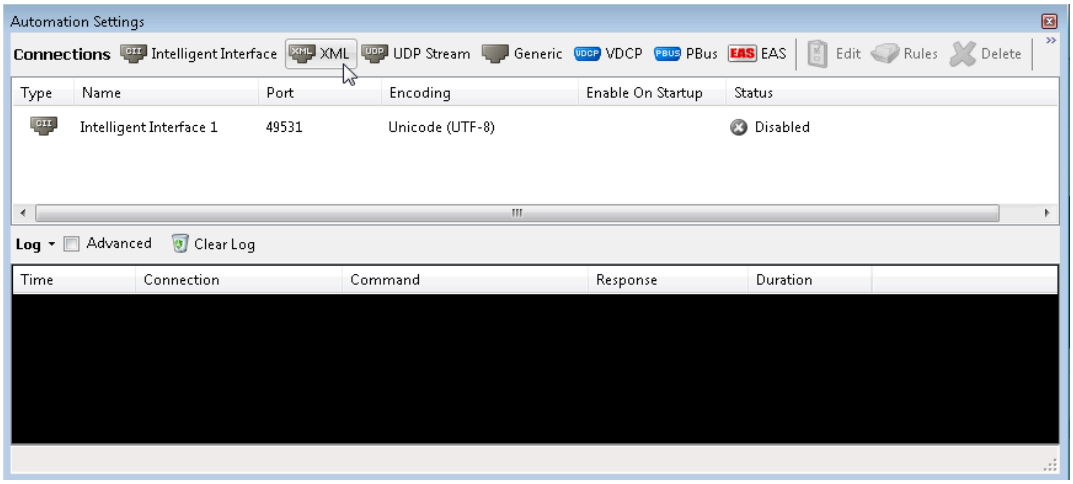
Initializing Prime XML Automation Control

To make use of the Prime automation control an endpoint must be created in the Prime designer to in order to listen for incoming connections. This is done by using the Automation Settings form.

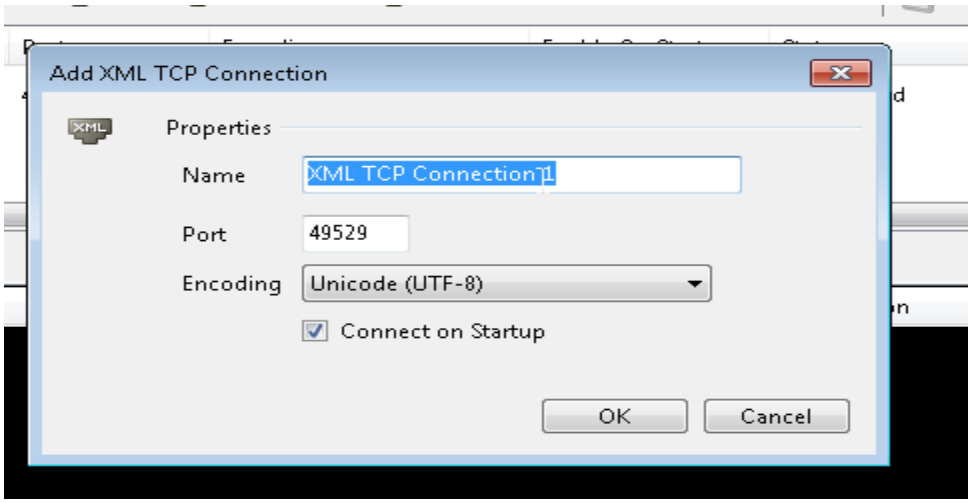
1. To open the Automation Settings form navigate to the “Config” menu strip item. After selecting “Config” select the “Automation..” item from the dropdown list.



2. Once the Automation Settings form is open choose the XML icon at the top of the form.



3. Once the XML option is selected a new form will open allowing the user to edit the values associated with the connection. For the XML automation connection the properties that are configurable are Name, Port, Encoding and whether or not the connection should start when Prime does.



4. Once the user presses OK, the connection will be created and be listening for incoming connections on the Port number assigned. An external application can then issue XML commands to Prime. A list of implemented commands are defined below.

Anatomy of an XML Automation Command

XML automation commands contain the needed information to complete an automation command in Prime. The majority of commands need to be able to access a channel or a clip player. This action will need to be completed before the command to continue. Each xml element node of a command string corresponds to a method call in prime. The methods are executed in sequential order based on the configuration of the command. This section will discuss the specific commands and how they are represented in XML format in Prime 2.0.

Each command must contain a <command>...</command> node container. Additionally, an id attribute can be associated with the top level command node for logging and debug purposes. The id attribute is optional.

Ex.) <command>...</command> or <command id=42>...</command>

Below are commands defined for Prime 2.0.

Scene Commands

Load Scene

The load scene command loads the scene on the channel specified in the FindChannel node of the command string. If the channel is not found Prime will load the scene on the default channel.

```
Ex.) <command id = 1>
        <Channel index=”1”>
          <Load scene=”SomeScene” />
        </ Channel>
     </ command>
```

Play Scene

The play scene command plays the scene from the channel specified in the FindChannel node of the command string. If the channel is not found Prime will try to play the scene on the default channel.

```
Ex.) <command id = 1>
        <Channel index=”1”>
          <Play scene=”SomeScene” />
        </ Channel>
     </ command>
```

Transfer Scene

The transfer scene command issues a stop to a playing scene. The scene is found on the channel provided by the FindChannel node. If the scene was not found on the specified channel the default channel is used.

```
Ex.) <command id = 1>
        <Channel index=”1”>
          <Transfer scene=”SomeScene” />
        </ Channel>
     </ command>
```

Clear Scene

The clear scene command closes the scene found on the channel provided by the FindChannel node. If the scene was not found on the specified channel the default channel is used.

```
Ex.) <command id = 1>
        <Channel index=”1”>
          <Clear scene=”SomeScene” />
        </ Channel>
     </ command>
```

Set Value(Scene)

The set value command changes scene object values to the provided value. The command can set one or multiple values. To do so, the command must first find the channel, the scene and the scene object provided in the command string. If all are found and the property exists the value of the object property will be changed to the value provided.

```
Ex.) <command id = 1>
        <Channel index=”1”>
          <Scene name=”SomeScene” />
            <Object name=”Text 1”>
              <SetValue property=”Text” value=”Hello World” />
            </ Object>
            <Object name=”Image 1”>
              <SetValue property=”File” value=”C:\Test.png” />
            </ Object>
          </ Scene>
        </ Channel>
      </ command>
```

Clip Commands

Load Clip

The load clip command loads the scene on the to the clip player specified in the FindClipPlayerByIndex node of the command string. If the channel is not found Prime will return an error message indicating that clip player couldn’t be found.

```
Ex.) <command id = 1>
        <ClipPlayer index=”1”>
          <LoadClip scene=”SomeScene” />
        </ ClipPlayer>
      </ command>
```

Play Clip

The play clip command plays the scene on the to the clip player specified in the FindClipPlayerByIndex node of the command string. If the channel is not found Prime will return an error message indicating that clip player couldn't be found.

```
Ex.) <command id = 1>
        <ClipPlayer index="1">
          <PlayClip scene="SomeScene" />
        </ ClipPlayer>
      </ command>
```

Clear Clip

The clear clip command removes the scene from the clip player specified in the FindClipPlayerByIndex node of the command string. If the channel is not found Prime will return an error message indicating that clip player couldn't be found.

```
Ex.) <command id = 1>
        <ClipPlayer index="1">
          <ClearClip scene="SomeScene" />
        </ ClipPlayer>
      </ command>
```

Pause Clip

The pause clip command pauses the scene specified in the FindClipPlayerByIndex node of the command string. If the channel is not found Prime will return an error message indicating that clip player couldn't be found.

```
Ex.) <command id = 1>
        <ClipPlayer index="1">
          <PauseClip scene="SomeScene" />
        </ ClipPlayer>
      </ command>
```

Set Value(Clip)

The set value command changes clip object values to the provided value. The command can set one or multiple values. To do so, the command must first find the clip player, the clip and the clip object provided in the command string. If all are found and the property exists the value of the object property will be changed to the value provided.

```
Ex.) <command id = 1>
        <ClipPlayer index=”1”>
          <Clip name=”SomeClip” />
            <Object name=”Text 1”>
              <SetValue property=”Text” value=”Hello World” />
            </ Object>
            <Object name=”Image 1”>
              <SetValue property=”File” value=”C:\Test.png” />
            </ Object>
          </ Clip>
        </ ClipPlayer>
      </ command>
```

Miscellaneous Commands

Get Thumbnail

The get thumbnail command return a thumbnail of either the channel or the clip player provided by the command string. It's size is determined by the width and height parameters provided in the command.

```
Ex.) <command id = 1>
        <Channel index="1">
          <GetThumbnail width="1920" height="1080"  />
        </ Channel>
      </ command>

      <command id = 1>
        <ClipPlayer index="1">
          <GetThumbnail width="1920" height="1080"  />
        </ ClipPlayer>
      </ command>
```

Get Channel Count

Returns the number of channels available.

```
Ex.) <command id = 1>
        <GetChannelCount />
      </ command>
```

Get Clip Player Count

Returns the number of clip players available.

```
Ex.) <command id = 1>
        <GetClipPlayerCount />
      </ command>
```

Aliases

Some nodes in the XML commands can be aliased to reduce typing and make the XML more readable. The functionality for the commands stays the same though the xml fragment may look slightly different. Some examples aliases can be found below.

Alias Examples

1. Load Scene
    a. Long Form
        i. <command id = 1>
        <FindChannel index=”1”>
        <Load scene=”SomeScene” />
        </ FindChannel>
        </ command>
    b. Alias
        i. <command id = 1>
        <Channel index=”1”>
        <Load scene=”SomeScene” />
        </ Channel>
        </ command>
2. Play Clip
    a. Long Form
        i. <command id = 1>
        <FindClipPlayerByIndex index=”1”>
        <PlayClip scene=”SomeScene” />
        </ FindClipPlayerByIndex>
        </ command>
    b. Alias
        i. <command id = 1>
        <ClipPlayer index=”1”>
        <PlayClip scene=”SomeScene” />
        </ ClipPlayer>
        </ command>

3. SetValue(scene)
    a. Long Form
        i. <command id = 1>
<FindChannel index=”1”>
<FindScene name=”SomeScene” />
<FindObject name=”Text 1”>
<SetValue property=”Text” value=”Hello World” />
</ FindObject>
<FindObject name=”Image 1”>
<SetValue property=”File” value=”C:\Test.png” />
</ FindObject>
</ FindScene>
</ FindChannel>
</ command>
    b. Alias
        i. <command id = 1>
<Channel index=”1”>
<Scene name=”SomeScene” />
<Object name=”Text 1”>
<SetValue property=”Text” value=”Hello World” />
</ Object>
<Object name=”Image 1”>
<SetValue property=”File” value=”C:\Test.png” />
</Object>
</ Scene>
</ Channel>
</ command>

4. SetValue(clip)
    a. Long Form
        i. <command id = 1>
<FindClipPlayerByIndex index=”1”>
<FindClip name=”SomeScene” />
<FindObject name=”Text 1”>
<SetValue property=”Text” value=”Hello World” />
</ FindObject>
<FindObject name=”Image 1”>
<SetValue property=”File” value=”C:\Test.png” />
</ FindObject>
</ FindClip>
</ FindClipPlayerByIndex>
</ command>

    b. Alias
        i. <command id = 1>
<ClipPlayer index=”1”>
<Clip name=”SomeScene” />
<Object name=”Text 1”>
<SetValue property=”Text” value=”Hello World” />
</ Object>
<Object name=”Image 1”>
<SetValue property=”File” value=”C:\Test.png” />
</Object>
</ Clip>
</ ClipPlayer>
</ command>

*Note: Currently only FindXXXX command nodes are aliased. Future Prime version may introduce new aliases as well as a shorthand command structure.

XML Automation Command Responses

All commands are accompanied by a response that represents the success or failure of the command that was issued. Upon completion of the command a response is created and sent back to the endpoint that initiated the command. A command can complete due to successful completion of all methods represented in the command structure, failure, should any of the methods fail when being called, or by the automation system being issued an unknown command.

Responses With No Return Value

Most commands do not have a return value therefore the responses to these commands will basically only indicate the success or failure of the command. A successful command will only indicate success, while a failed command will indicate fail and potentially an exception message if the method failed ungracefully. Examples of command responses with no return value are displayed below.

Success

```
<response id="2">
  <return method="Load">
    Success
  </return>
</response>
```

Failure

General Failure

```
  <response id=4>
    <error method=Play>
      Play failed. Arguments: scene:Test2
    </error>
  </response>
```

Unknown Method Failure

```
  <response id=4>
    <error method=UnknownMethod>
      The command UnknownMethod does not exist.
```

```
  </error>
 </response>
```

Exception Failure

```
<response id=4>
  <error method=GetThumbnail>
   Unhandled Exception: System.NullReferenceException: Object reference not set to an
   instance of an object.
  </error>
 </response>
```

Responses With Return Values

There are a certain few commands that return information upon request. These command responses will contain a similar structure as the responses without a return value however the result of the command will be contained with the response text. The commands implemented to return data from Prime are listed below.

GetChannelCount

Returns the number of channels configured in the playout configuration.

```
 <response id="1">
  <return method=”GetChannelCount">
   8
  </return>
 </response>
```

GetClipPlayerCount

Returns the number of clip players configured in the playout configuration.

```
 <response id="1">
  <return method=”GetChannelCount">
   8
  </return>
 </response>
```

GetValue

Returns the value for specified property of and object in the scene.

```
<response id="1">
  <return method=”GetValue property=Text">
    Hello World!
  </return>
</response>
```

GetThumbnail

Returns a base64 encoded string representing a thumbnail created from the scene.

```
<response id="2">
  <return method="GetThumbnail">
    Base64StringGoesHere
  </return>
</response>"
```

ABOUT US

Chyron is ushering in the next generation of storytelling in the digital age. Founded in 1966, the company pioneered broadcast titling and graphics systems. With a strong foundation built on over 50 years of innovation and efficiency, the name Chyron is synonymous with broadcast graphics. Chyron continues that legacy as a global leader focused on customer-centric broadcast solutions. Today, the company offers production professionals the industry's most comprehensive software portfolio for designing, sharing, and playing live graphics to air with ease. Chyron products are increasingly deployed to empower OTA & OTT workflows and deliver richer, more immersive experiences for audiences and sports fans in the arena, at home, or on the go.

CONTACT SALES

EMEA • North America • Latin America • Asia/Pacific
+1.631.845.2000 • sales@chyron.com

Chyron